

## DATA PROCESSING APPARATUS AND DATA PROCESSING METHOD

### CROSS REFERENCE TO RELATED APPLICATIONS

5 This application claims priority to Japanese Patent Application No. JP 2000-357208 filed in November 24, 2000, the disclosure of such application being herein incorporated by reference to the extent permitted by law.

### 10 BACKGROUND OF THE INVENTION

#### 1. Field of the Invention

The present invention relates to a data processing apparatus and a data processing method for dealing with an incidental interruption of power supply in a system.

#### 15 2. Description of Related Art

Conventionally, if an incidental interruption of power supply occurs in a system controlled by a computer, recovering has been done either by restarting from a completely initial stage or by hibernation, in which contents of the memory during operation before the interruption (or, blackout) are read out and execution of operations are resumed.

However, restarting presents a problem that carrying out all processes, such as detection of devices, initialization and the like typically required long time to perform. On the other hand, 25 in the case of the hibernation, memory information is written out irrespective of a program under execution and amount of busy memory, thus causing a problem that it took long time to store the memory condition as data as well as to recovering it. Moreover, hibernation has a problem that it cannot be executed 30 in the background during processing and it cannot cope with the incidental interruption of power supply.

Conversely, as a method of coping with the incidental interruption of power supply such as a sudden blackout or the like, there is a method of detecting a low voltage condition and 35 then instructing an application to end processing. However,

such method requires a circuit for detecting the voltage status condition. Thus, such method has a problem in which even if the circuit for detecting the voltage condition is installed, a system software program cannot assure whether all data can be stored in time until the interruption of power supply after the generation of the low voltage condition.

However, as this method having been used as a measure against power supply interruption, so that for a case of recovering the system, this method is carried out perfectly similarly to restarting or rebooting. Hence, this has a problem that it takes a long time for recovering to an operable condition.

Therefore, the present invention has been conceived so as to provide a data processing apparatus and a data processing method, in which, even if an incidental interruption of power supply occurs in a system, it can immediately return back to a status immediately before the interruption of power supply.

## SUMMARY OF THE INVENTION

A first preferred embodiment of the present invention includes a data processing apparatus provided with a central processing unit and a memory including a driver controlling an operation for writing to and reading from a recording medium, wherein when there is a request for a status-storing process from a component, a dependency relation of the component and/or stored data is stored as a snap shot file in the recording medium, and when there is a request for a status recovering process, a status of the component is recovered in accordance with the snap shot file stored in the recording medium.

A second preferred embodiment of the present invention provides a data processing method including a central processing unit and a memory, wherein an operation for writing to and reading from a recording medium is controlled by a driver, and when there is a request for a status-storing process from a component, a dependency relation of the component and/or stored data are stored as a snap shot file in the recording medium, and when there is a request for a status recovering process, a

status of the component is recovered in accordance with the snap shot file stored in the recording medium.

When the status-storing process is outputted from the component, the dependency relation of the component and/or the stored data are recorded as the snap shot file in the recording medium. In the case of the occurrence of power supply interruption, the status recovery of the component is carried out from the snap shot file recorded in the recording medium, in accordance with the dependency relation of the component.

A third preferred embodiment of the invention provides a computer program for performing the steps of the data processing method of the second embodiment. Also, another embodiment describes a storage medium storing a software program in computer-readable form, in which the software program enables a computer to perform the steps of the data processing method of the second embodiment.

#### BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects, features and advantages of the present invention will become more apparent from the following description of the presently preferred exemplary embodiments of the invention taken in conjunction with the accompanying drawings, in which:

Fig. 1 is a block diagram showing a system to which the present invention can be applied;

Fig 2 is a flowchart describing an example of an initialization of a component according to a preferred embodiment of the present invention;

Fig. 3 is a schematic diagram describing a relation between modules for a system status-storing process according to a preferred embodiment of the present invention;

Figs. 4A and 4B are examples of formats of a snap shot file according to a preferred embodiment of the present invention;

Fig. 5 is a flowchart describing an example of a status-storing process according to a preferred embodiment of

the present invention;

Fig. 6 is a schematic diagram describing a relation between modules for a system status recovering process according to another preferred embodiment of the present invention; and

Fig. 7 is a flowchart describing an example of a status recovering process according to a preferred embodiment of the present invention.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

An embodiment of the present invention will be described below with reference to the drawings. By the way, the same reference symbols are given to those having the same functions in the respective drawings, in order to avoid duplication of explanation. Fig. 1 shows a schematic block diagram of an embodiment of a system to which the present invention is applied. A system shown in Fig. 1 is provided with a CPU (Central Processing Unit) 1, a RAM (Random Access Memory) 2, a device 3, a non-volatile memory 4 and a bus 5.

The RAM 2 is a memory on which writing/reading operation can be freely performed. The device 3 is, for example, a serial device, a graphic device or the like. The non-volatile memory 4 is a non-volatile memory device, such as a flash ROM, a disc or the like. Also, the non-volatile memory 4 may be a detachable device (media), such as a flash memory card or the like, namely, a so-called IC card. The CPU 1 is connected to the RAM 2, the device 3 and the non-volatile memory 4 through the bus 5, which leads to a condition that they can be communicated with each other. Also, the bus 5 may be a network.

The control of initialization of a component provided for each function of the present invention will be described below with reference to a flowchart shown in Fig. 2. At a step S1, when a component provided for each function (hereafter, referred to simply as a component) such as an application, a

device, a driver or the like is activated, an internal variable thereof is initialized. At step S2, a dependency relation between the components is recorded in a status-storing database by the component itself. At a time of a system status-storing processing as described later, the component recorded in this status-storing database is regarded as a target for the status-storing operation. In this way, the flowchart shown in Fig. 2 shows an initialization operation at a time of booting of an application, a device driver or the like.

A system status-storing process will be described below with reference to Fig. 3. Fig. 3 shows the relation between modules to be executed by the CPU 1, which exist on a memory that can be accessed by the CPU 1. A request for the status-storing process is outputted from a certain application among applications 14 to a check point manager 12 in this module 11. A request for an execution of a snap shot output to the component, such as the application 14, a device driver 15 and the like, is outputted to the check point manager 12. An order or sequence in which this request is outputted is based on a sequence recorded in a status-storing database 13 in this module 11. The sequence recorded in this status-storing database 13 represents the dependence relation between the components.

When each component receives the request for the snap shot output, a function existing in a particular address for each component is read out and the component status is outputted as a snap shot data through the check point manager 12 to a non-volatile memory driver 16. When the output of the snap shot data for each component is ended, all the output snap shot data are used as snap shot files shown in Figs. 4A, 4B, and the non-volatile memory driver 16 stores them in a recording medium (not shown). The non-volatile memory driver 16 supports an operation for writing to and reading from various recording media, for example, a non-volatile memory card. Then, the process returns back to the application 14 outputting the request for the status-storing process.

A tag of a component name (character string) shown in

0998920-111901  
1001

5 Figs. 4A, 4B is given to the snap shot file, which is stored in the recording medium by the non-volatile memory driver 16, for example, as a combination of an index file and a data file. Fig. 4A shows an example of a format of the index file, and Fig. 4B shows an example of a format of the data file. As shown in Fig. 4A, the index file stores therein a component name, and an offset position in a data file of a data corresponding to the component name. As shown in Fig. 4B, the snap shot data outputted by each component is stored in the data file, in connection with one file.

15 As mentioned above, the status-storing process of the system is activated (booted) at arbitrary timing during the execution of the application or for each cycle specified by the system or the application. Once the status-storing process is activated, a next request for the status-storing process is kept until an end of that process.

20 The snap shot file to be recorded in the recording medium by the non-volatile memory driver 16 is recorded in accordance with the sequence recorded in the status-storing database 13. Thus, the dependency relation between the components is reflected.

25 It will be described below with reference to a flowchart of an example of the status-storing process of the system in Fig. 5. At a step S11, the snap shot file is initialized. At a step S12, the status-storing database 13 is opened. At a step S13, it is judged whether or not the component targeted for the status storing operation exists or is available. In accordance with the sequence recorded in the status-storing database 13, if it is judged that the component targeted for the status storing operation still exists, the control proceeds to a step S14. If it is judged that the component targeted for the status storing operation does not exist, the control proceeds to a step S16.

30 At step S14, the index data is outputted from the component targeted for the status storing operation through the check point manager 12 to the non-volatile memory driver 16. At a step S15, the snap shot data is outputted from the

35

component targeted for the status storing operation through the check point manager 12 to the non-volatile memory driver 16.

At a step S16, the snap shot file stored in the recording medium by the non-volatile memory driver 16 is closed. At a  
5 step S17, the status-storing database 13 is closed.

A system status recovering process will be described below with reference to Fig. 6. Fig. 6 shows a relation between the modules to be executed by the CPU 1, which exist on the memory that can be accessed by the CPU 1. A booting unit 17  
10 of the system outputs a request for the status recovering process to the check point manager 12. In accordance with this request for the status recovering process, the check point manager 12 requests the non-volatile memory driver 16 to read the snap shot file. As mentioned above, the dependency relation between the  
15 components are reflected in the snap shot file. The check point manager 12 retrieves a component of an application device driver 18 having its status recovered, from the component name stored in the snap shot file, and firstly reads the component as necessary, and carries out booting.

Then, an address of a function to be called from a symbol information held by the component is obtained through the non-volatile memory driver 16 from the recording medium, and the function is called from the address. The function to be  
20 called exists for each component in a particular address. Those processes are executed for all the components stored in the snap shot file. By the way, at this time, the dependency relation between the components is recorded in the status-storing database 13 within this module 11 by the component itself. The content recorded in this status-storing database 13 is accessed at  
25 the time of a further request for the status-storing process.  
30

By the way, the status recovering process can be carried out not only at the time of the booting of the system but also an arbitrary timing. However, once the status recovering process is activated, a next request for the status recovering process is  
35 kept standing by until an end of the previous status recovering process.

10647-0269660

5 An example of the system status recovering process will be described below with reference to a flowchart of Fig. 7. At a step S21, the snap shot file stored in the recording medium by the non-volatile memory driver 16 is opened. At a step S22, the status-storing database 13 is opened. At a step S23, the index data stored in the recording medium by the non-volatile memory driver 16 is read in. At a step S24, it is judged whether or not the component that needs to be read is ended, in accordance with the stored index data. If it is judged in accordance with the index data that the component that needs to be read is not still ended, the control proceeds to a step S25. If it is judged that all the components that need to be read are ended, the control proceeds to a step S29.

15 At step S25, the stored snap shot data is read from the recording medium through the non-volatile memory driver 16. At a step S26, the component is read in accordance with the read in snap shot data, and it is activated. At a step S27, the status recovering process is carried out. Actually, the address of the function to be read out is obtained from the symbol information held by the component. Then, the function is called from the address. The function to be called exists for each component in a particular address. At a step S28, the component itself and the dependency relation between components are recorded in the status-storing database 13 within this module 11 by the component itself. Then, the control returns back to the step S23.

25 At a step S29, the snap shot file stored in the recording medium by the non-volatile memory driver 16 is closed. At a step S30, the status-storing database 13 is closed.

30 In this preferred embodiment of the invention, even if the application operates on the system, the status-storing operation can be executed in the background without an operator operating the system taking notice of such operation.

35 This preferred embodiment of the present invention can be applied to any apparatus if it has the configuration shown in Fig. 1. For example, this embodiment can be applied to a portable



telephone, PDA (Personal Digital Assistant), a notebook PC or the like.

In the preferred embodiment present here, if the recording medium in which the snap shot file is stored by the non-volatile memory driver 16, for example, an IC card is inserted into a PC different from a PC in which the snap shot file is stored, and the status recovering process is then carried out, the PC into which the IC card is inserted can be used in an environment completely equivalent to that of the PC in which the snap shot file is stored.

According to the preferred embodiments of present invention, it is possible to carry out booting of a device in accordance with the dependency relation of the device, when the power supply is again turned on, and it is further possible to recover the original status at a high speed.

Moreover, according to the preferred embodiments present invention, the data required to recover a status of a certain component is shunted or recovered by the component itself. Thus, recovering can be attained at a minimum necessary small amount of data.

Also, according to the preferred embodiments of the present invention, as the information for the recovery is stored in a portable memory device, if the memory device is connected to another apparatus, the status can be recovered at a high speed.

Moreover, according to the present invention, the status can be recovered not only at the time of the booting of the system but also an arbitrary timing.

Although the present invention been described in its preferred form with a certain degree of particularity, obviously many changes, variations, combinations and sub-combinations are possible therein. It is therefore to be understood that any modifications will be practiced otherwise than as specifically described herein without departing from the scope of the present invention.